

Bottleneck Paths

time limit per test: 1 second (3 seconds for Python)
memory limit per test: 500 megabytes
input: standard input
output: standard output
points: 100 · problem code: bottleneck

You are given a connected undirected graph with n vertices and m weighted edges. For every pair of vertices, find a path that **minimizes the maximum edge weight** along the path.

In other words, if P ranges over all paths from u to v , the bottleneck is

$$\min_P \max_{e \in P} w(e).$$

Input

The first line contains two space-separated integers n and m .

Each of the next m lines contains three integers u , v , and w , denoting an undirected edge between u and v with weight w . There may be multiple edges between the same pair of vertices, and there may be self-loops.

Vertices are numbered 1 through n .

Output

Print n lines. The i^{th} line should contain n space-separated integers $a_{i,1}, a_{i,2}, \dots, a_{i,n}$, where $a_{u,v}$ is the bottleneck between u and v . For every u , print $a_{u,u} = 0$.

Constraints

- $1 \leq n \leq 100$
- $1 \leq m \leq \frac{n \cdot (n-1)}{2}$
- $1 \leq u, v \leq n$
- $1 \leq w \leq 10^9$

Example 1

Input

```
4 5
1 2 5
1 3 9
2 3 1
4 2 3
3 4 2
```

Output

```
0 5 5 5
5 0 1 2
5 1 0 2
5 2 2 0
```

Explanation

Between 1 and 3, the direct edge has weight 9, but the path $1 \rightarrow 2 \rightarrow 3$ has maximum edge $\max(5, 1) = 5$, which is better. Between 2 and 4, the path $2 \rightarrow 3 \rightarrow 4$ has maximum edge 2, which beats the direct edge of weight 3.